Doctoral Thesis Proposal

A language-level approach to distributed computation with weak synchronization and its application to cloud and edge computing environments.

Christopher Meiklejohn
Université catholique de Louvain
christopher.meiklejohn@gmail.com

November 1, 2015

Contents

1 Proposed System

1.1 Abstract

Given the nature of large-scale "Internet of Things" [1] and mobile applications[1], it is impractical to assume that the traditional client-server architecture will be able to scale the amount of data generated at the edge.

The traditional model for edge computing focuses around client-server interactions: clients located at the edge either generate large amounts of immutable data or mutate replicated shared state and periodically perform synchronization of this state with the server. Two different approaches exist here: either the clients at the edge stream data back to a central data center representing a time series of mutations for a given data object, or the clients periodically synchronize their state with either the client or the server storing the "source of truth" for a given data item. Each of these approaches have their own set of drawbacks.

In the stream-based approach, clients generate streams of events that must be consumed by a central data center for processing. These event streams typically require the delivery of all events in causal order, to the central data center in order to guarantee the "correctness" of computations using that data as input. [3, 4] This technique requires enough computational power for consuming all values across all streams for all connected devices, and enough bandwidth to support the transmission and consumption of these streams. This technique also requires that devices remain connected to facilitate the transmission of these streams, which is a challenging task given the cost of operating the antenna to available power on the units.

In the alternative approach, either clients or servers store the "source of truth" for each data item. Mutations are performed locally at each client, and state is periodically synchronized with the server. This is the typical approach taken by many web and mobile applications [5], where a majority of the computation is performed at the client. While this approach is correct when shared state is minimized, if clients can concurrency act on replicated copies of this state, concurrency control becomes a problem. In the event of concurrent operations to shared state, most of these systems need to resort to some method of arbitration: for instance, with two conflicting copies of the same object resulting from concurrent operations by two clients, we may choose the "winning" copy of the data item based on temporal time.

Both of these approaches result in two fundamental challenges for the developer of distributed applications: **distribution** and **nondeterminism**. Inherent in distributed computation, the application developer needs to explicitly decide where to place the "source of truth" for each data item and where that data should be replicated based on where computations involving it occur. Given replicated data is essential to fault-tolerance and coordinating changes to replicated data reduces availability of systems, application developers must decide what "consistency model" needs to be used for each type of computation: weaker models allow the system to make progress when only part of the

[1] Rovio, developer of the popular "Angry Birds" game franchise reported that during the month of December 2012 they had 263 million active users. This does not account for users who play the game on multiple devices, which is an even larger number of devices requiring some form of shared state in the form of statistics, metrics, or leaderboards. [2]

system is online or partial results are available. Finally, weakening coordination in systems ultimately results in nondeterminism given items can make progress independent of others, leaving the application developer to deal with explicitly reducing observable nondeterminism.

As such, one area of improvement for application developers is to provide a programming model and language-based abstractions for dealing with the problems of **distribution** and **nondeterminism**.

1.2 Related Work

Previous work on Conflict-Free Replicated Data Types (CRDTs) [6, 7] and the research platform that supports them, SwiftCloud [8], presents a data abstraction supporting deterministic resolution of concurrent modifications to a single data item. This data abstraction allows clients to track state that is periodically synchronized with the server, while guaranteeing deterministic convergence of replicated state once all updates are delivered to all nodes in the system. This work has also been very successful in industry in the Riak database [9]. Twitter's Summingbird [3] system also uses similar methods to allow concurrent processing of elements in a stream of items given the commutativity properties that some computations observe. Work on directed diffusion [10] and digest diffusion [11] have explored optimized models for performing aggregation of information across large-scale sensor networks.

Work on using CRDTs to build large-scale distributed computations has been previously explored as well. Navalho et al. [12, 13] have provided specifications for building CRDTs that perform computations and present a system designed for computing aggregates using CRDTs. Conway et al. [14] have designed a programming model that combines monotonic logic and convergent modules to support correct, coordination-free programming. Meiklejohn and Van Roy [15, 16] have designed a programming model that use CRDTs as the primary data abstraction for distributed computation.

1.3 Research Goals

The goal of this research is to provide a **declarative** way to design **distributed**, fault-tolerant applications that do not contain observable **nondeterminism**. These applications should be able to be placed at arbitrary locations in the network: mobile devices, "Internet of Things" hardware, or personal computers. Applications should be tolerant to arbitrary message delays, duplication and reordering: these are first-class requirements of distributed computations over unreliable networks. When writing these applications, developers should not have to use traditional concurrency control or synchronization mechanisms such as mutexes, semaphores, or monitors: the primitive operations for composition in the language should yield "determinisic-by-construction" applications.

1.4 Description Of Work

Our goals rely on us pushing the boundaries of what can be done with minimal amounts of synchronization, given the cost of performing synchronization across a large number of geographically distributed clients.

We break our work up into several phases:

1.4.1 Functional Programming Language

We continue to expand our previous work on Lasp [15, 16], a functional programming language build using CRDTs as the primary data abstraction. We continue to extend our Lasp work to operate over richer types of data structures: for example, sets that can contain only a bounded number of elements efficiently and optimized set representations that generate less garbage. We grow the language into a higher-order language, and realize several of our example use cases as Lasp programs that live fully within the language and do not rely on components from the language hosting it (such as Erlang, in the case of our original prototypical implementation). We investigate methods for invariant preservation between two data items and how to write correct applications while minimizing coordination as much as possible.

1.4.2 Distributed Runtime

We explore optimized methods of distribution for Lasp applications. We investigate methods for generating the most optimal distribution for applications given an example program. For example, given a user application and requirements for placement for the given data items, can the system automatically derive the most optimum placement for the remainder of the data items given the requirements of efficiency and fault-tolerance. We explore how to efficiently and correctly aggregate information in applications containing a large number of clients, such as "Internet of Things" style sensor networks and mobile applications running on smartphones.

1.4.3 Evaluation

We evaluate different models of distribution for various Lasp applications with example use cases from industry partners such as Rovio Entertainment, Basho Technologies and Machine Zone. We attempt to identify where the declarative functional programming abstraction breaks down: are there cases where it is too impractical to derive an optimal distribution strategy for an application without the application developer explicitly programming in the required constraints? We evaluate the network scalability of composed CRDTs in large-scale geographically distributed applications. We identify the bounds of what types of applications can be developed without application developer specified coordination mechanisms.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] "263 Million Monthly Active Users In December," http://www.rovio.com/en/news/blog/261/263-million-monthly-active-users-in-december, accessed: 2015-02-13.

[3] O. Boykin, S. Ritchie, I. O'Connell, and J. Lin, "Summingbird: A framework for integrating batch and online mapreduce computations," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1441–1451, 2014.

[4] S. Khare, K. An, A. Gokhale, and S. Tambe, "Functional Reactive Stream Processing for Data-centric Publish/Subscribe Systems."

[5] "Ember Data source code repository," https://github.com/emberjs/data, accessed: 2015-08-03.

[6] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A comprehensive study of convergent and commutative replicated data types," INRIA, Tech. Rep. RR-7506, 01 2011.

[7] ——, "Conflict-free replicated data types," INRIA, Tech. Rep. RR-7687, 07 2011.

[8] M. Zawirski, A. Bieniusa, V. Balegas, S. Duarte, C. Baquero, M. Shapiro, and N. Preguiça, "Swiftcloud: Fault-tolerant geo-replication integrated all the way to the client machine," *arXiv preprint arXiv:1310.3107*, 2013.

[9] R. Brown, S. Cribbs, C. Meiklejohn, and S. Elliott, "Riak DT map: a composable, convergent replicated dictionary," in *Proceedings of the First Workshop on Principles and Practice of Eventual Consistency*. ACM, 2014, p. 1.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 56–67.

[11] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*. IEEE, 2003, pp. 139–148.

[12] D. Navalho, S. Duarte, N. Preguiça, and M. Shapiro, "Incremental stream processing using computational conflict-free replicated data types," in *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*. ACM, 2013, pp. 31–36.

[13] D. Navalho, S. Duarte, and N. Preguiça, "A study of CRDTs that do computations," in *Proceedings of the First Workshop on Principles and Practice of Consistency for Distributed Data*. ACM, 2015, p. 1.

[14] N. Conway, W. R. Marczak, P. Alvaro, J. M. Hellerstein, and D. Maier, "Logic and lattices for distributed programming," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 1.

[15] C. Meiklejohn and P. Van Roy, "Lasp: a language for distributed, eventually consistent computations with CRDTs," in *Proceedings of the First Workshop on Principles and Practice of Consistency for Distributed Data*. ACM, 2015, p. 7.

[16] ——, "Lasp: A language for distributed, coordination-free programming," in *Proceedings of the 17th Symposium on Principles and Practice of Declarative Programming (PPDP 2015)*. ACM, Jul. 2015.